

Project Report

Team ID	PNT2022TMID04798
Project Name	CONTAINMENT ZONE ALERTING APPLICATION

1. INTRODUCTION:

1.1 Project Overview:

Currently there are several research works undergoing in the country to prevent Covid-19 cases from rising. Previously our country was importing medical kits like PPE (Personal Protection Kits), mask from outside, but now it has been successful in developing these kits. Along with taking initiatives to fight this disease, our country has also taken steps to make people aware of the disease. The news and media have a great part in creating this awareness by informing the public about the preventive measures that can keep them away from infection. Awareness among the people to carry out all the preventive measures can immensely help to reduce spread of the virus. The country has created containment zones throughout the cities wherever Covid-19 cases have been reported to prevent further spread of the virus. These containment zones have been kept isolated from the outside public to ensure no contamination occurs outside. After more than 2 months of the lockdown, the government has relaxed some of the lockdown rules and has permitted reopening of government offices, bus and other road transportation facilities and shopping markets. People can move inside the city for work and other purposes. But the containment zones are still being kept isolated, and new containment zones are being formed wherever Covid-19 cases have been reported. These zones are highly contagious as droplets with virus coughed out from an unscreened asymptomatic patient can travel up to 8 m (Bahl et al. 2020). Though these containment zones are guarded by policemen, still there remains a chance that people might unknowingly step into them. In this situation where people can move in the city, these containment zones pose a risk of infection to these city dwellers. Therefore, informing people about the location of the containment zones can help them bypass and avoid these zones and thereby reduce the chance of community transmission. In this paper, we focus on developing a mobile based application to provide information regarding the Covid-19 containment zones in West Bengal. The application further tracks the user's location and provides notification alert if the user has entered a containment zone. The application also provides daily Covid-19 case statistics to the users to keep them updated. The application is developed on Android SDK and uses Firebase Cloud Firestore to store the location data. Android's geofencing client is used to create geofences around the containment zones and notification manager is used to provide notifications. The application also uses RESTful web services to show the Covid-19 cases in West Bengal. We have tested our application with different users in different locations across West Bengal and it works efficiently and is able to attain our target.

1.2 Purpose:

The Android application shows the location of the containment zones to the users. It also notifies the user when he or she trespasses the boundary of a containment zone or stays in the containmentzone.



2. LITERATURE SURVEY:

2.1 Existing problem:

People doesn't have proper knowledge about containment zones since they do change daily and hard to keep updated and if they are not updated properly, they will lead to wide spread of disease.

2.2 Reference:

PAPER 1:

TITLE: Tracking the Covid zones through geo-fencing technique

AUTHOR NAME: Anto Arockia Rosaline R ,Lalitha R ,Hariharan G ,Lokesh

PUBLICATION YEAR: 2017

DESCRIPTION:

Following the tracking of a suspicious person, the geo-fenced layer is mapped out in the vicinity, and the virtual perimeter is then employed for the subsequent trapping procedure. As soon as the Covid monitoring team updates this geo-fenced layer, the public can view it. The idea of creating a virtual perimeter region is known as geo-fencing. Effective containment zone monitoring is made possible by this virtual perimeter monitoring technology. By utilising an automated system based on wireless infrastructure, it lowers operational costs. Additionally, it promptly alerts the law enforcement to find the offenders. As a result, it facilitates the inspection of containment areas and the monitoring of those who disobey governmental regulations. Users can receive updates from the Covid team on the alert zone. The Covid team has a number of modules for suspect tracking, hotspot fencing, etc. The Covid team must seek a service from the service network provider in the case of suspect tracking, and following authorization, they will offer the coordinates. According to our telecommunication legislation, it is illegal to share data; nonetheless, exchanging personal information without the individual's knowledge via any means is occasionally allowed with governmental approval for investigative purposes.

PAPER 2:

AUTHOR NAME: Geofencing 2.0: Taking Location-based Notifications to the Next Level

PUBLICATION YEAR: 2016

DESCRIPTION:

Sandro Rodriguez Garzon Bersant Deva The basic Android application that served as the prototype Geofencing client was used. This client is primarily responsible for carrying out the geofencing server's ongoing location update strategy. This must be accomplished with little energy consumption because the Geofencing client is located on a mobile device. We made the decision to employ the low energy Geofencing features of the Android operating system to keep an eye on the safety zone. As a result, a safety zone is considered as a single circular geofence with a required exit on the mobile device. However, they discovered that there was occasionally a significant lag time between leaving the safety zone and receiving a notification from the system about the leave. In order to address this issue, a specific amount of the safety zone's radius is decreased. While the safety zone and how it is implemented have a significant impact on overall energy consumption, it is also important to make the right choice when it comes to a placement mechanism. In order to reduce power consumption without compromising the necessary position precision, they used a device-based smart combination of various positioning mechanisms introduced by. By temporarily deactivating placement when a device is not in motion, the Geofencing client also makes use of cutting-edge mobile sensing capabilities integrated into the Android operating system's activity recognition unit. Mobile users who live close to a geo-border fence's will find this to be of particular utility. If the Geofencing server notifies the Geofencing client about a geonotice, the notification will appear right away.

PAPER 3

TITLE: Development of An Android Application for Viewing Covid19 Containment Zones Alerting.

AUTHOR NAME: India Ranajoy Mallik, Amlan Protim Hazarika, Sudarshana Ghosh Dastidar, Dilip Sing & Rajib Bandyopadhyay

PUBLICATION YEAR: 2019

DESCRIPTION:

The World Health Organization has declared the outbreak of the novel coronavirus, Covid-19 as pandemic across the world. With its alarming surge of affected cases throughout the world, lockdown, and awareness (social distancing, use of masks etc.) among people are found to be the only means for restricting the community transmission. In a densely populated country like India, it is very difficult to prevent the community transmission even during lockdown without social awareness and precautionary measures taken by the people. Recently, several containment zones had been identified throughout the country and divided into red, orange and green zones, respectively. The red zones indicate the infection hotspots, orange zones denote some infection and green zones indicate an area with no infection. This paper mainly focuses on development of an Android application which can inform people of the Covid-19 containment zones and prevent trespassing into these zones. This Android application updates the locations of the areas in a Google map which are identified to be the containment zones. The application also notifies the users if they have entered a containment zone and uploads the user's IMEI number to the online database. To achieve all these functionalities, many tools, and APIs from Google like Firebase and Geofencing API are used in this application. Therefore, this application can be used as a tool for creating further social awareness about the arising need of precautionary measures to be taken by the people of India.

PAPER 4:

TITLE: Aarogya Setu

AUTHOR NAME: National Informatics Centre, Ministry of Electronics & Information Technology, Government of India.

PUBLICATION YEAR: 2014

DESCRIPTION:

The most popular containment zone alert application among the options currently in use in India is called Aarogya Setu. The Indian government created a mobile application to link the public with crucial health services. Its primary features include geo-location-based COVID19 data, user risk status, automatic contact tracing using Bluetooth, and much more. The movement of an infected individual is tracked using Bluetooth and GPS technology, and the system notifies the public of the locations the infected person has visited while designating those locations as vulnerable ones. It employs cellular triangulation to determine a person's location in the absence of GPS technology. While Aarogya Setu can track down contacts and notify those who have come into touch with someone who has COVID-19, it also actively keeps track of quarantine or containment zones and alerts users who enter them. The Terms of Use and Privacy Policy must be accepted at the time of registration when installing the application on any Android or iOS mobile device, and ongoing use of the application denotes continued acceptance. Name, age, sex, occupation, phone number, overseas travel within the previous 28–45 days, and whether the user is a smoker are all pieces of information that the app gathers. This data is kept on a server that is under the jurisdiction of the Indian government. It is hashed and sent to the user's mobile application along with a special digital ID (DID). The user is recognised using the DID. In order for the user's mobile phone to exchange information with another device that has the app when it gets within range, the Bluetooth and GPS services must be turned on. Their individual IDs, along with the time and GPS location, are kept on the two phones when two users come into close proximity. The format in which this data is kept is encrypted. Only after a person tests positive is it posted to the government-controlled servers of the app.

2.3 Problem Statement Definition:

PROBLEM STATEMENT 1:



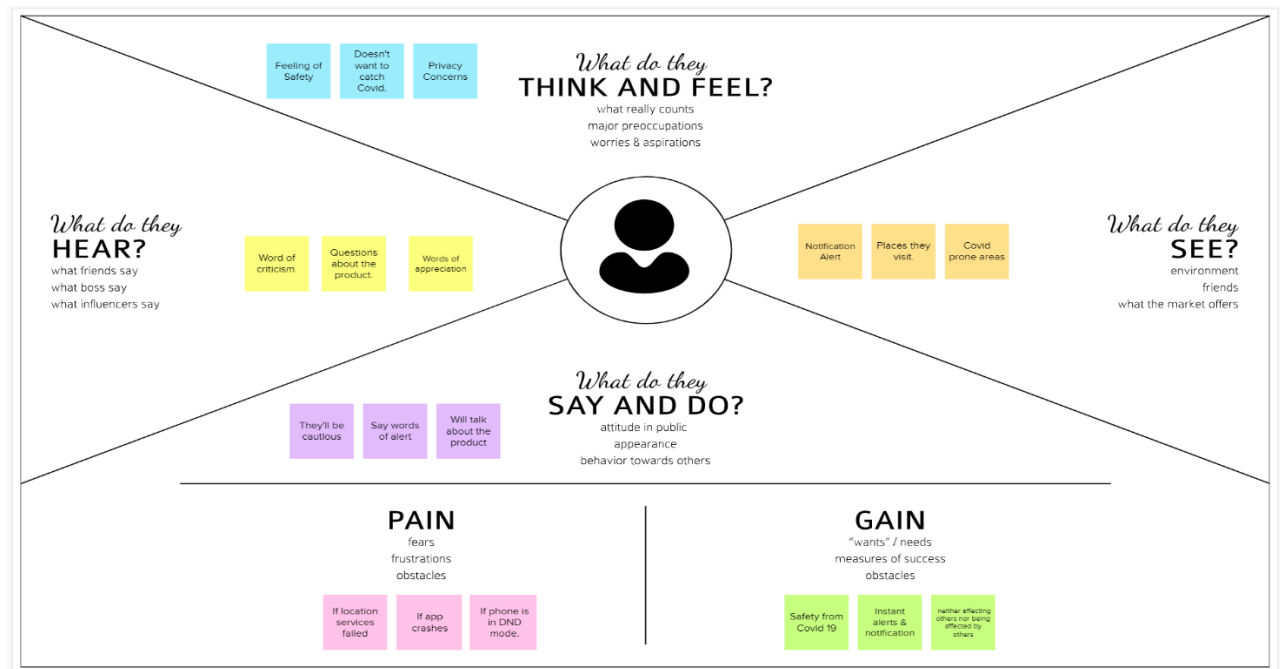
PROBLEM STATEMENT 2:



3. IDEATION AND PROPOSED SOLUTION:

3.1 Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



3.2 Ideation & Brainstorming:

template



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- 10 minutes to prepare
- 1 hour to collaborate
- 2-8 people recommended

[Share template feedback](#)

➔

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

- Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.
- Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.
- Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) ➔

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

problem

The organizers wanted to create an online platform where students could learn together and share their knowledge. They needed to be able to find relevant content, track their progress, and collaborate with others. The platform also needed to be easy to use and accessible to all students.

Key rules of brainstorming

To run a smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.



Need some inspiration?

See a finished version of this template to inspire your team.

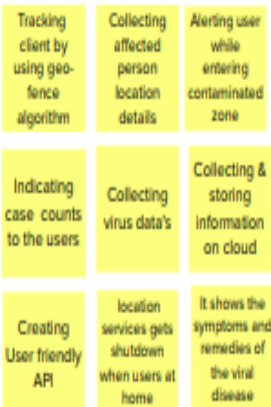
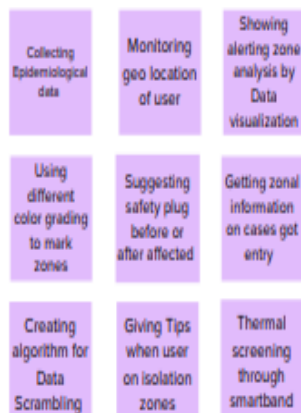
[Open example](#) ➔

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

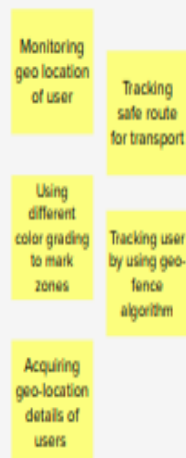
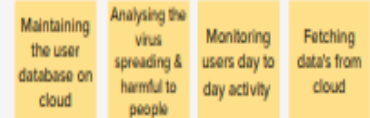
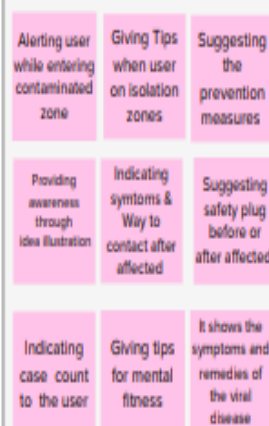
ASHOK N**ASWIN S****Dharanya T****BHUPESHKUMAR K**

3

Group Ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

30 minutes

LOCATION**DATA COLLECTION****DATA USAGE****ALERTS****SPECIAL FEATURES**

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes



+

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

- Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.
- Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

- Strategy blueprint**
Define the components of a new idea or strategy.
[Open the template →](#)
- Customer experience journey map**
Understand customer needs, motivations, and obstacles for an experience.
[Open the template →](#)
- Strengths, weaknesses, opportunities & threats**
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
[Open the template →](#)

[Share template feedback](#)



3.3 Problem Solution:

S.NO	PARAMETER	DESCRIPTION
1.	Problem Statement (Problem to be solved)	To solve people's science about the pandemic on the outside of their environment and giving people warning on possible danger.
2.	Idea / Solution description	Creating app that "Tracks live location of the User and alerts User when user trespasses into the contaminated zone or stays in the containment zone". Connecting hospitals to get "Medical data of patients". Using Blockchain technology for location and data encryption" to protect data getting into wrong hands.
3.	Novelty / Uniqueness	It connects people virtually thereby it provides alert messages and suggests remedies for the pandemic or virus spreading out there.
4.	Social Impact / Customer Satisfaction	Build people watch out! on pandemic out there. Propose people restorative on medicine for emergency assistance on viruses that spreading. Exposing case count and riskiness of viruses through daily broadcast.
5.	Business Model (Revenue Model)	Designing a test utensil that does basics test on finding out infected people on their own. Introducing premium plan that monitors user health by connecting app with their smart band. Giving services at home for a premium member
6.	Scalability of the Solution	In this model user provided with the medical services through online and giving nescience to people by giving restoratives on medicines and monitoring user movements on pandemic zones and alert before they affected.

3.4 Problem Solution fit:

Problem-Solution Fit canvas			Purpose / Vision	Version:
Define CS, fit into CL	1. CUSTOMER SEGMENT(S) CS People who made travel from one place to another are our customers.	6. CUSTOMER LIMITATIONS CL <small>EG. BUDGET, DEVICES</small> Should avoid traveling Economy problem Financial instability	5. AVAILABLE SOLUTIONS AS <small>PLUSES & MINUSES</small> The app shows a containment zone but only for covid 19 virus that spread out. Safe route displaying app available that only show the safe route to travel but can not be used by all users.	Explore AS, differentiate
	2. PROBLEMS / PAINS PR <small>+ ITS FREQUENCY</small> Moving to some new location without knowing the environmental conditions. If they are stuck on some problem find out the solution to get out of it.	9. PROBLEM ROOT / CAUSE RC Nescience about the pandemic. Not maintaining the proper prevention measures. Not getting proper guidance on medicinal inputs.	7. BEHAVIOR BE <small>+ ITS INTENSITY</small> The behavior of the virus can not be calculated exactly. Its intensity or vulnerability is calculated according to its incubation time.	
Identify strong TR & EM	3. TRIGGERS TO ACT TR People in this society got a fear of the fake news spreading through social media that triggers people to act wrongly in this pandemic situation.	10. YOUR SOLUTION SL Creating an app that connects people and doctors through the cloud platform. Connecting hospitals and getting case information by giving separate login to the doctors and Creating an algorithm that analyzes the data given by the hospitals and gives out the output of harmful viruses that spread. Creating a user-friendly interface that provides alert messages when the user enters the contaminated zone	8. CHANNELS of BEHAVIOR CH ONLINE Online users will get information about the virus that spreading out there.	Extract online & offline CH of BE
	4. EMOTIONS EM <small>BEFORE / AFTER</small> People would feel safe and happy because they came to know about the pandemic spreading out and they were aware before they were infected.		OFFLINE Offline users will get the service provided by the company	

 Problem-Solution fit canvas is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
 Designed by Daria Nepriakhina / ideaHackers.nl - we tailor ideas to customer behaviour and increase solution adoption probability.

 [IdeaHackers.nl](https://ideaHackers.nl)

4. REQUIREMENT ANALYSIS:

4.1 Functional Requirement:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail. Registration through mobilenumber.
FR-2	User Confirmation	Confirmation via Email. Confirmation via OTP.
FR-3	Authentication	It checking the confirmation of the password.
FR-4	Business rule	For subscriber's we give first 3 day's free trail. For un subscriber's the user needs to watch some advertisement for knowing the zone alert for first 3 day's.

4.2 Non-Functional requirements:

Following are the non-functional requirements of the proposed solution.

NR No.	Non-Functional Requirement	Description
NFR-1	Usability	Providing recommendation link by using customer preference .
NFR-2	Security	The software team will issue some strong security code for the user's.
NFR-3	Reliability	The database update process must rollback all related updates when any update fails.
NFR-4	Performance	The loading speed of the server is quick and fast.

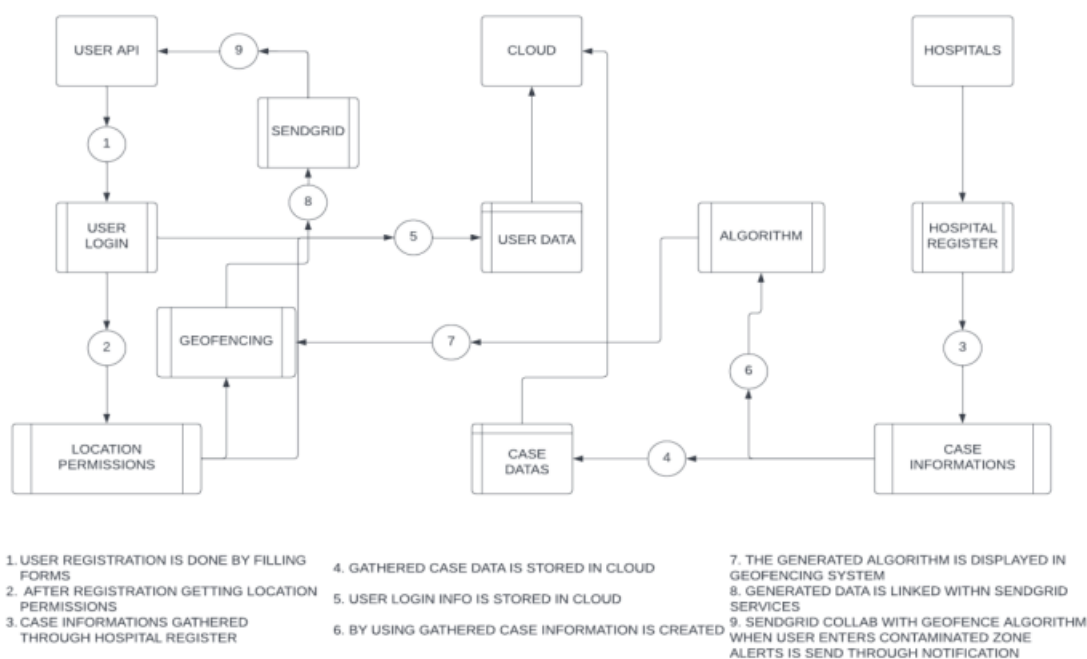
5. PROJECT DESIGN:

Data Flow Diagrams

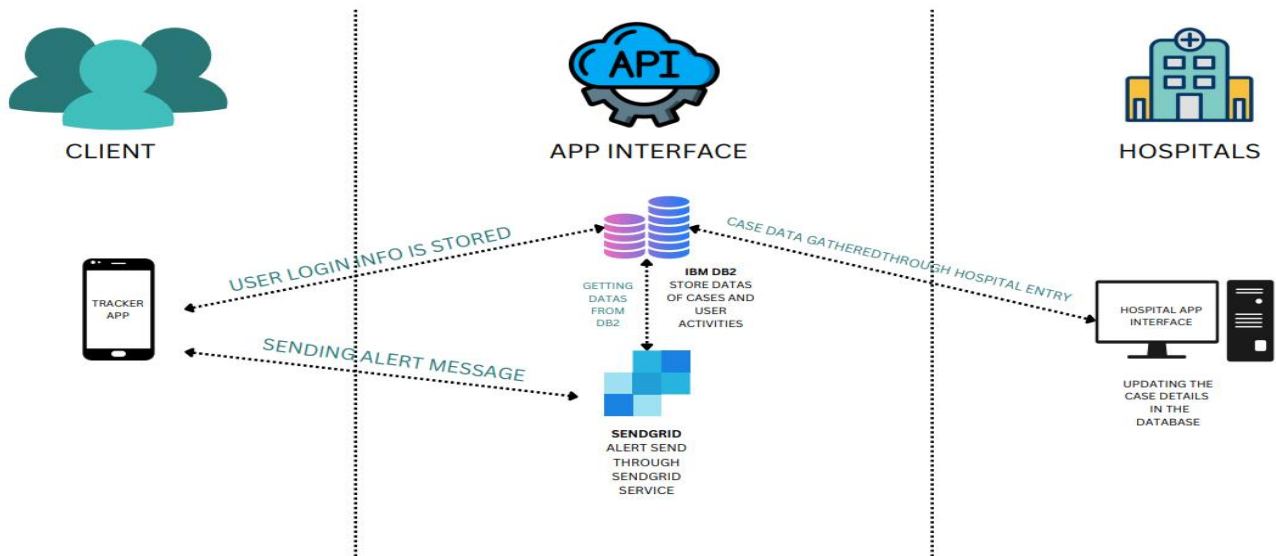
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically.

It shows how data enters and leaves the system, what changes the information, and where data is stored.

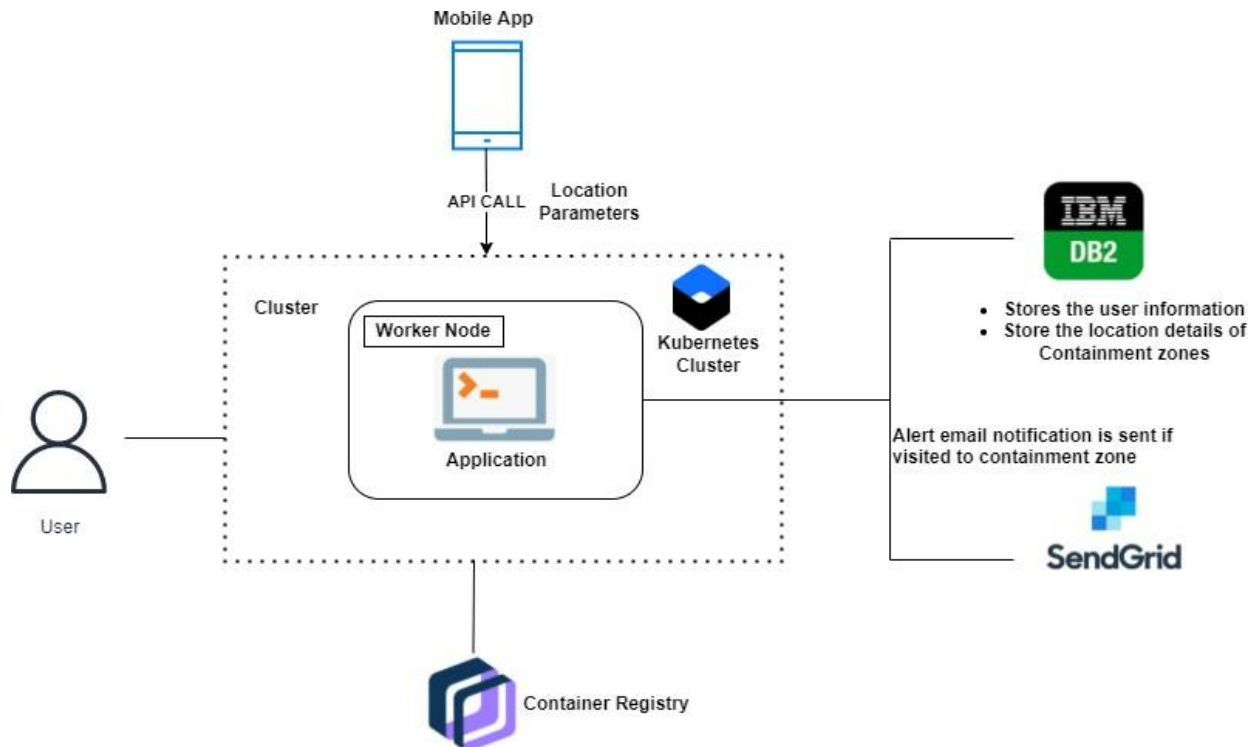
5.1 Data flow diagram:



5.2 SOLUTION ARCHITECURE:



5.2 TECHNICAL ARCHITECTURE:



5.3 User Stories:

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Login	Registration (web and android)	USN-1	I can register for the application by entering my email and password	I can control my online account and dashboard.	Medium	Sprint-1
Sign Up	Registration (web and android)	USN-2	I will receive a confirmation email once I have registered for the application	I can handle the waste collection.	High	Sprint-1
Services	Dashboard	USN-3	need to give permission to access my location	I can take the shortest path to reach the waste filled route specified.	Medium	Sprint-2
Services	Service	USN-4	I need to differentiate the containment zones	I can collect the trash, pull it to the truck, and send it out.	Medium	Sprint-3
Data collection	Service	USN-5	. I need to alert the user when they enter the containment zone through the notification	All of these processes are under my control.	High	Sprint-4

6. PROJECT PLANNING & SCHEDULING:

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the selected project & gathering information by referring the, technical papers, research publications etc.	19 OCTOBER 2022
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	18 OCTOBER 2022

Ideation	List the by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	18 OCTOBER 2022
-----------------	--	-----------------

6.1 Sprint Delivery & Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
1	Registration	USN-1	USER: I can register for the application by entering my email and password	3	High	Ashok Aswin BhupeshKumar Daranya
		USN-2	USER: I will receive a confirmation email once I have registered for the application	2	High	Ashok Aswin BhupeshKumar Daranya
	Login	USN-3	USER: I can log into the application by entering my email & password	3	High	Ashok Aswin BhupeshKumar Daranya

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
2	Dashboard	USN-4	USER: I need to give permission to access my location	5	High	Ashok Aswin BhupeshKumar Daranya
		USN-5	USER: I can view the map with the containment zones	5	High	Ashok Aswin BhupeshKumar Daranya
	Service	USN-6	ADMIN: I need to update the containment zones.	5	High	Ashok Aswin BhupeshKumar Daranya

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
3	Service	USN-7	ADMIN: I need to differentiate the containment zones based on the intensity of infection.	3	Medium	Ashok Aswin BhupeshKumar Daranya
		USN-8	ADMIN: I need to provide precautionary measures when they travel.	3	Medium	Ashok Aswin BhupeshKumar Daranya
		USN-9	ADMIN: I need to provide information about the nearby hospitals	3	Low	Ashok Aswin BhupeshKumar Daranya

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
4	Service	USN-10	ADMIN: I need to alert the user when they enter the containment zone through email or SMS	5	High	Ashok Aswin BhupeshKumar Daranya
		USN-11	ADMIN: I need to provide medical recommendations by collaborating with hospitals.	3	Low	Ashok Aswin BhupeshKumar Daranya
	Data collection	USN-12	ADMIN: I need to store user details on the cloud	5	High	Ashok Aswin BhupeshKumar Daranya
		USN-13	ADMIN: I need to collect details about covid-19 cases from verified sources	5	High	Ashok Aswin BhupeshKumar Daranya

6.2. Sprint Delivery Schedule:

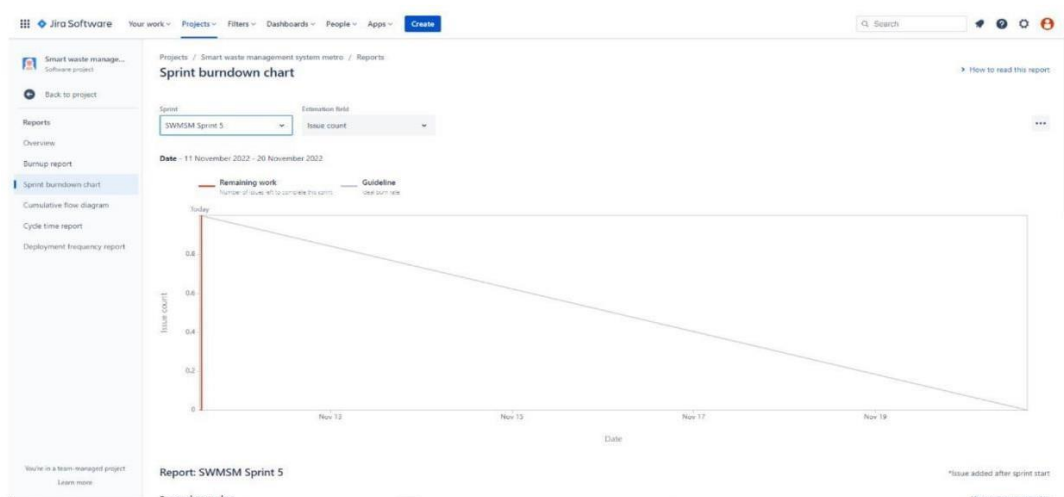
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)

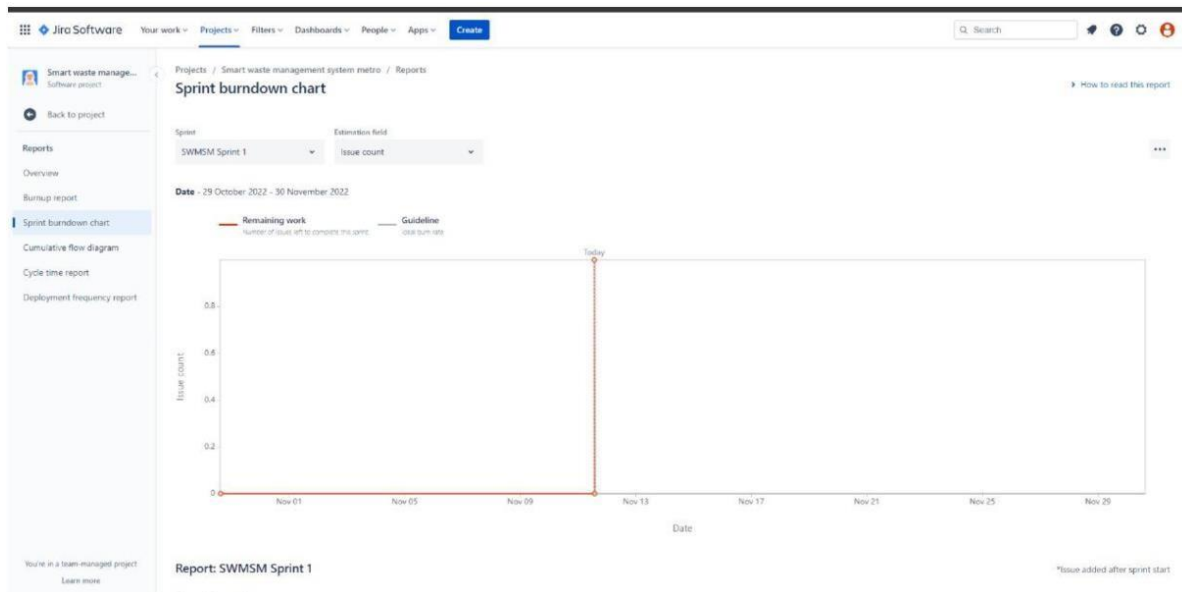
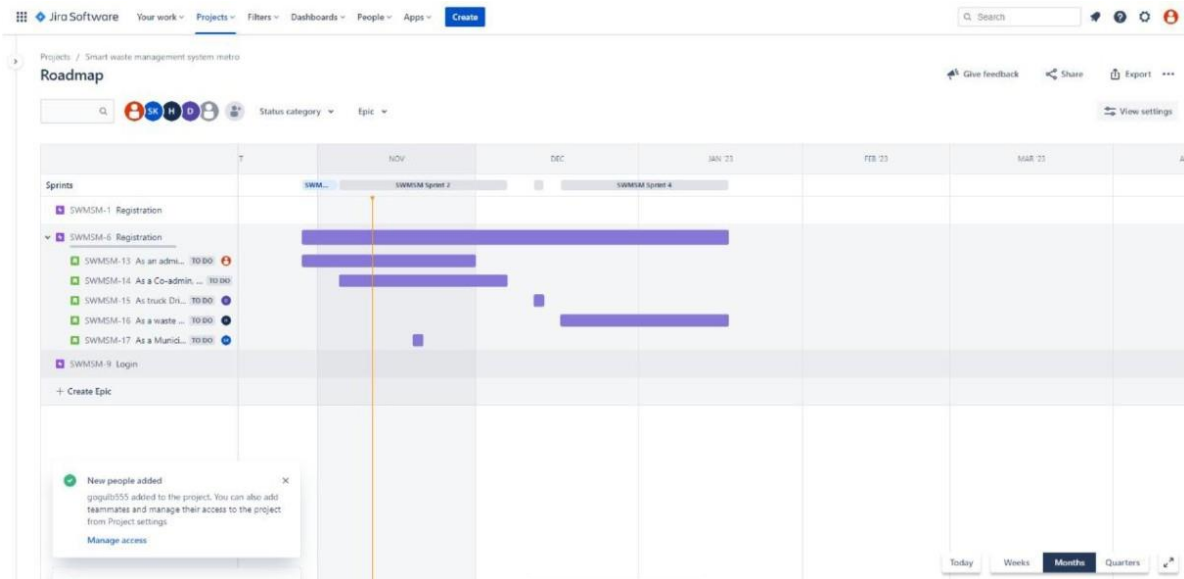
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

Velocity:

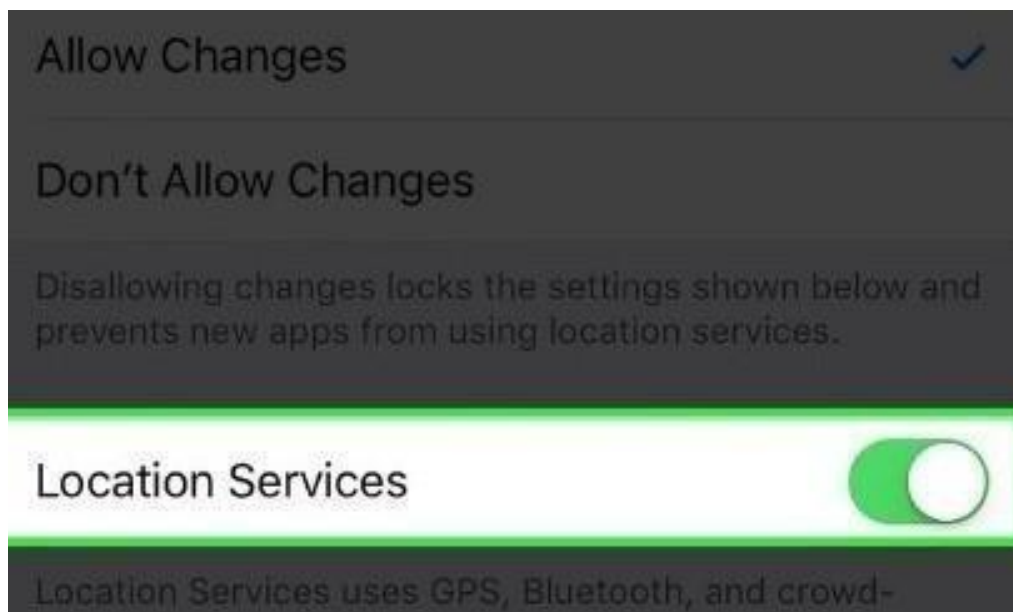
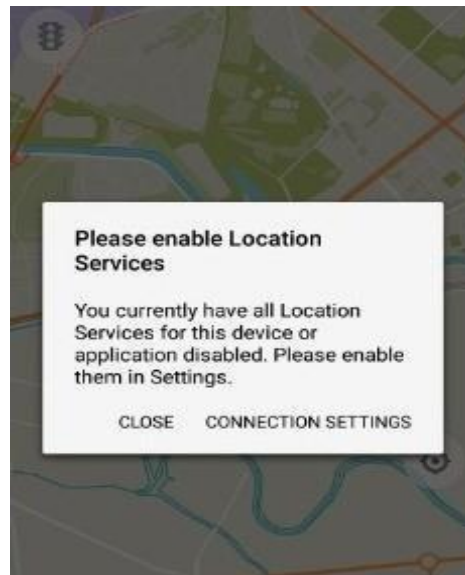
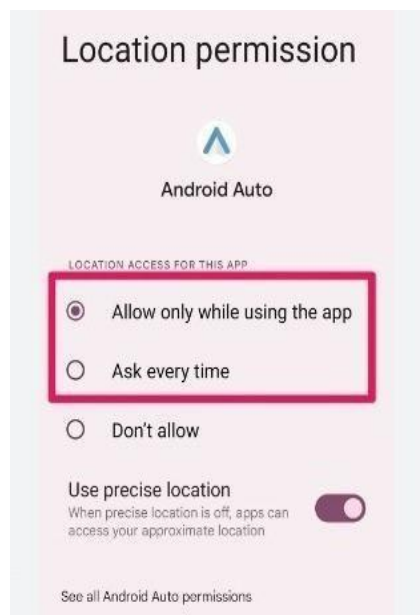
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

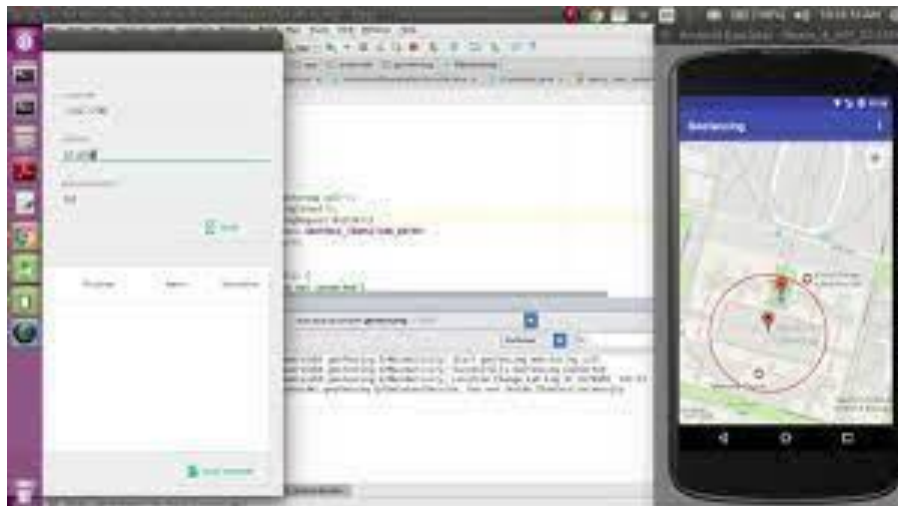
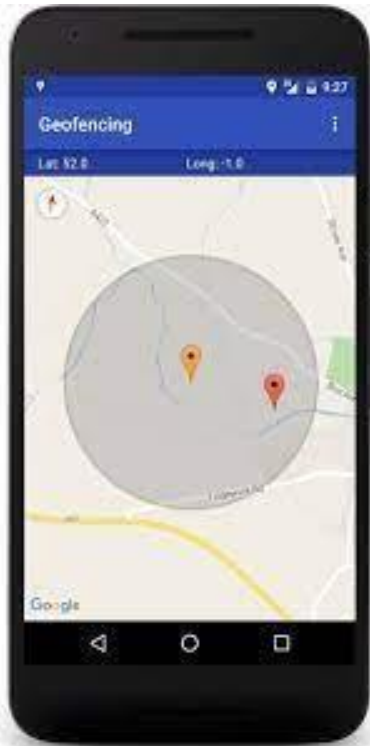
6.3 Reports from JIRA:**BURNDOWN CHART**



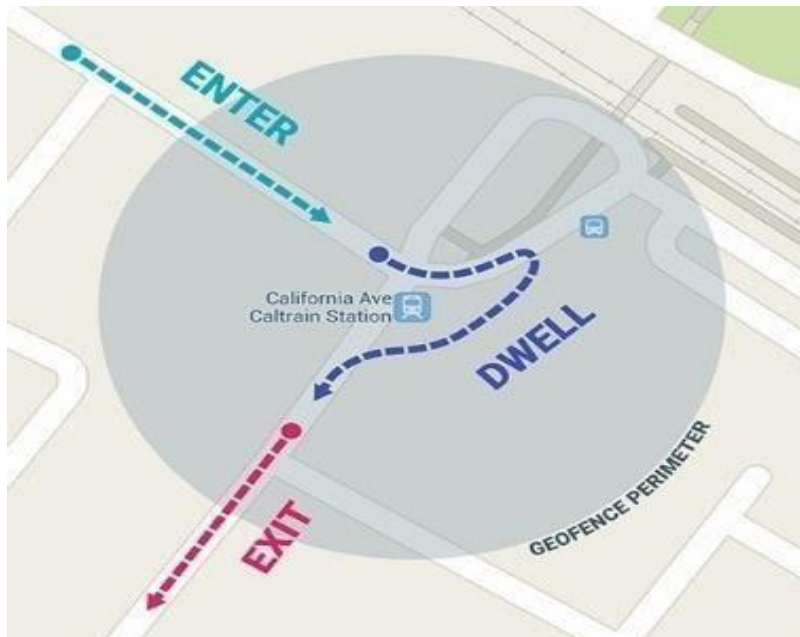
7. CODING & SOLUTIONING (Explain the features added in the project along with code)



GEOFENCE IN ANDROID APP :







8. RESULTS:

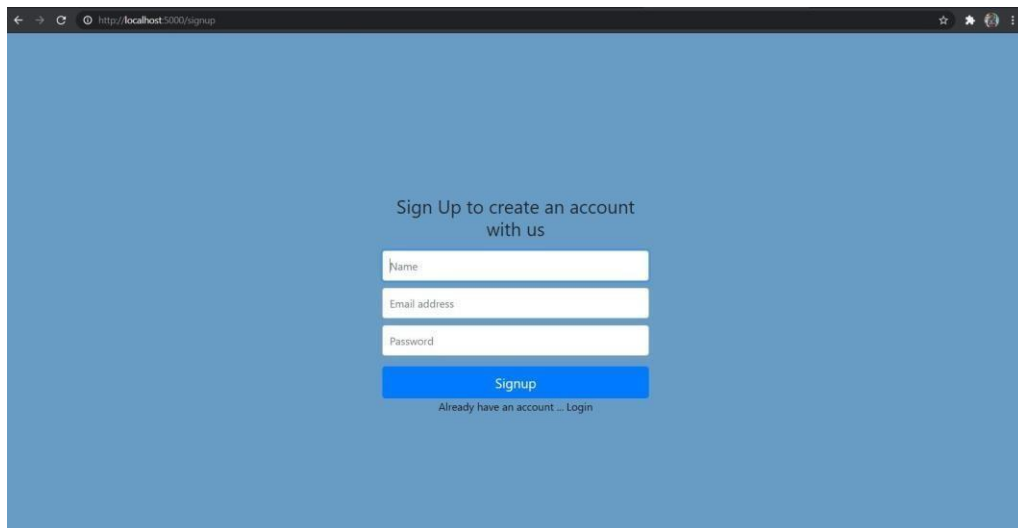
UI Interact with Application:

Admin App:

Login Page:

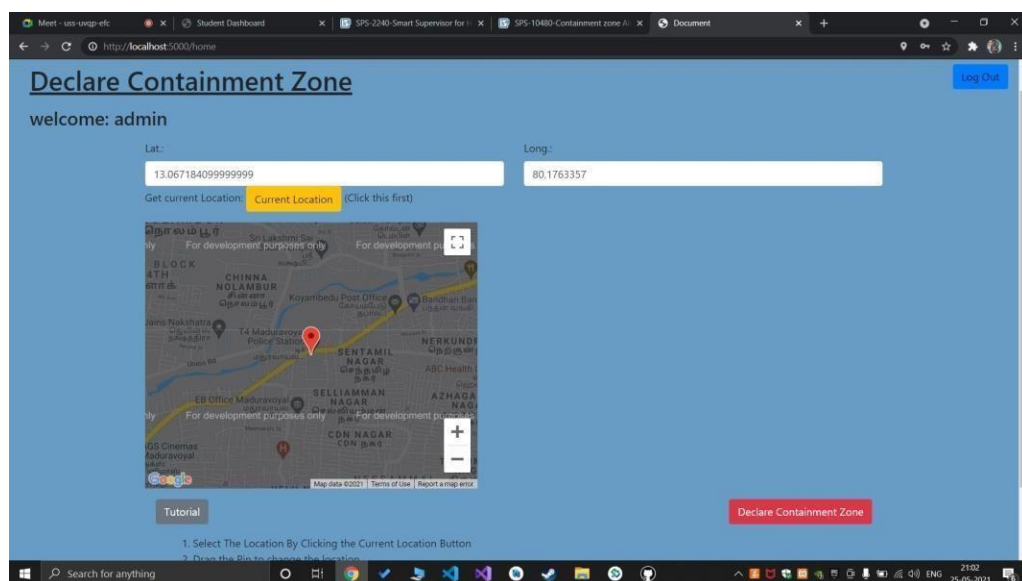
A screenshot of a web browser displaying a login page. The browser's address bar shows 'http://localhost:5000'. The page has a solid blue background. In the center, there is a white login form. The form contains the text 'Log In to add the location of the containment zone' above two input fields: 'Email address' and 'Password'. Below these fields is a blue button labeled 'Login'. At the bottom of the form, there is a link that says 'Don't have an account ... Create One'.

Register page:



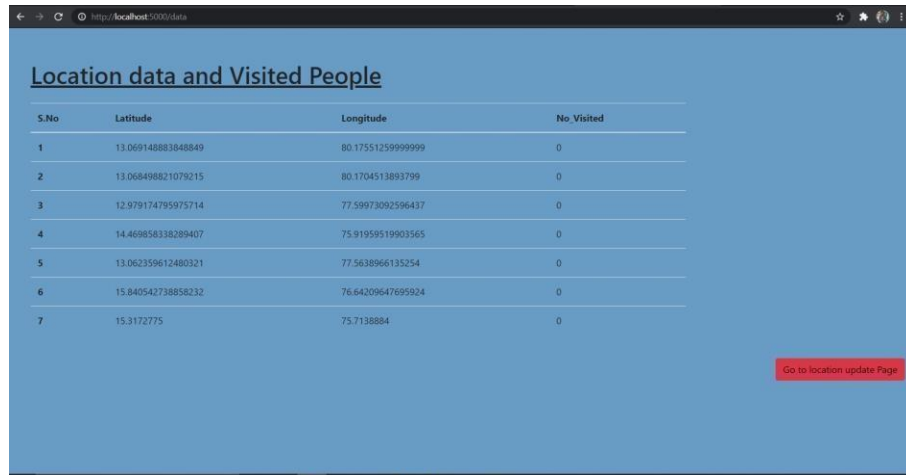
A screenshot of a web browser showing a registration page. The browser's address bar displays 'http://localhost:5000/signup'. The page has a solid blue background. In the center, the text 'Sign Up to create an account with us' is displayed. Below this text are three white input fields with black borders, labeled 'Name', 'Email address', and 'Password'. Underneath these fields is a blue button with the text 'Signup' in white. At the bottom of the form area, there is a link that says 'Already have an account ... Login'.

Home page:



A screenshot of a web browser showing a home page titled 'Declare Containment Zone'. The browser's address bar displays 'http://localhost:5000/home'. The page has a blue header with the title and a 'Log Out' button. Below the header, it says 'welcome: admin'. There are two input fields for 'Lat:' and 'Long:'. The 'Lat:' field contains '13.067184099999999' and the 'Long:' field contains '80.1763357'. Below these fields is a button labeled 'Current Location' with a tooltip that says '(Click this first)'. Underneath is a map showing a location in Chennai, India, with a red pin. Below the map is a 'Tutorial' button and a 'Declare Containment Zone' button. At the bottom, there are instructions: '1. Select the Location By Clicking the Current Location Button' and '2. Press the Pin to change the location.' The browser's taskbar at the bottom shows various icons and the system clock indicating 21:02 on 24-05-2021.

Location data page:

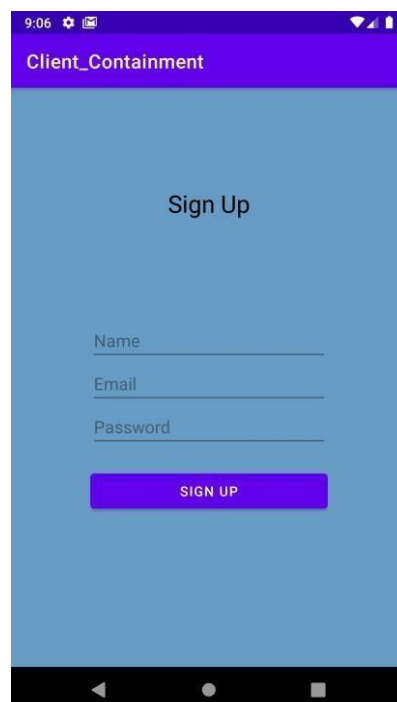


The screenshot shows a web browser window with the address bar displaying 'http://localhost:5000/data'. The page title is 'Location data and Visited People'. It contains a table with 4 columns: 'S.No', 'Latitude', 'Longitude', and 'No. Visited'. The table lists 7 entries. A red button labeled 'Go to location update Page' is located at the bottom right of the table area.

S.No	Latitude	Longitude	No. Visited
1	13.069148881048849	80.17551259999999	0
2	13.068498821079215	80.1704513893799	0
3	12.979174795975714	77.59973092596437	0
4	14.469858338289407	75.91959519903565	0
5	13.062359612480321	77.5638966135254	0
6	15.840542738858232	76.64209647695924	0
7	15.3172775	75.7138884	0

[Go to location update Page](#)

Client Application: Register screen:



The screenshot shows a mobile application interface with a purple header bar labeled 'Client_Containment'. The main screen is light blue and features the text 'Sign Up' in the center. Below this, there are three input fields labeled 'Name', 'Email', and 'Password'. At the bottom, there is a purple button labeled 'SIGN UP'.

9:06

Client_Containment

Sign Up

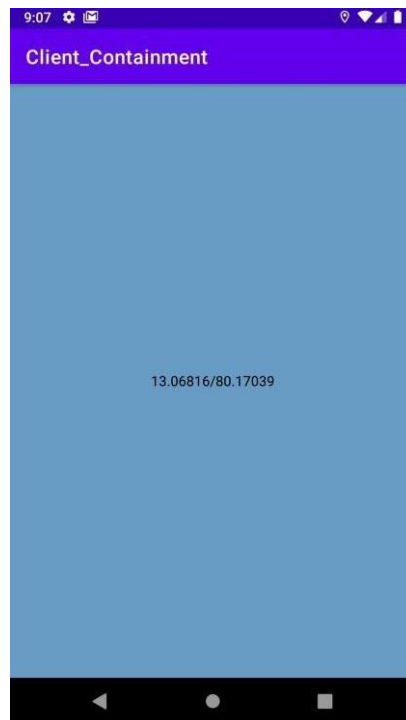
Name

Email

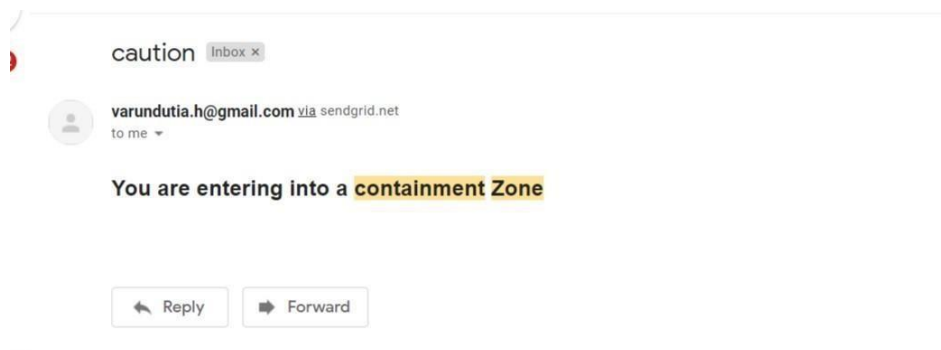
Password

SIGN UP

Current Location:



An Email will be sent to the registered mail id if the location is within 100 meters of the locations present in the admin app.



9. ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

- People can be alerted before entering containment zone.
- Further spread of virus can be reduced considerably.

DISADVANTAGES:

- Accuracy of application depends on the number of data given to the application.

- Application's accuracy is directly proportional to the number of data given to the application
- about the infected patients.

10. CONCLUSION:

This application is intended to provide information about containment zones in a particular region by alerting people, through continuous monitoring of an individual's location. Key benefits of

the application are monitoring people's activity and alerting them to their safety movements.

11. FUTURE SCOPE:

Although we tried to cover almost all of the aspects during our developmental phase, however we were forced to leave some aspects because of lack of time as well as monetary and other reasons. Just like in the field of software development where there are always some shortcomings and room for improvement our application can be enhanced further:-

- 1) The application can include various government organization to help act faster.
- 2) The dataset obtained from the application can be used for predictive analysis to determine prone areas and include special method for tackling the problem in those areas.
- 3) Emergency signal in case of network failure and internet connection loss.
- 4) Tackling victim's movements.
- 5) Improved Google positioning system's precision.
- 6) The client part of application can be integrated in a single intelligent device.

For analysis purpose, we could use machine learning (ML) algorithms as well as data mining applications. There is a sub branch of machine learning known as time series analysis (TSA), which could be used to predict and analyze the data obtained through this application. Time series analysis is used to predict crop production as well as sales in different quarter.

12. APPENDIX:

Source Code:

```
# Project : CONFINEMENT ZONE ALERTING APPLICATION Team ID :  
# PNT2022TMID04798
```

APP.PY:

```
from logging import  
error from flask import *  
from jinja2.utils import select_autoescape  
import bcrypt  
from flask_mysql import MySQL
```

```

import json
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail

# initialization
app = Flask(__name__)

# config
app.secret_key = "\x19Ts\xbe\xe7\x8c_\r\x12Q\x14\x13>q\xb7'WTH0\x9f\xe4\xec\xb1"
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'zone2'

mysql = MySQL(app)

# functions

def send_mail(email):
    print(email)
    message = Mail(from_email='aswinnkl710@gmail.com',
                    to_emails=email,
                    subject='caution', plain_text_content='Please Stay Safe',
                    html_content='<h2>You are entering into a containment Zone</h2>')

    try:
        sg = SendGridAPIClient(
            'SG.7BJDtQDIS8unH0r5_TufVQ.Ykpcz19QcqgcNwYZC3a0mNRPhGksG117YURqOTa2HL')
        response = sg.send(message)
        print(response.status.code)
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e)

def create_bcrypt_hash(password):
    # convert the string to bytes
    password_bytes = password.encode()
    # generate a salt
    salt = bcrypt.gensalt(14)
    # calculate a hash as bytes
    password_hash_bytes = bcrypt.hashpw(password_bytes, salt)
    # decode bytes to a string
    password_hash_str = password_hash_bytes.decode()
    return password_hash_str

```

```
def verify_password(password, hash_from_database):
    password_bytes = password.encode()
    hash_bytes = hash_from_database.encode()

    # this will automatically retrieve the salt from the hash,
    # then combine it with the password (parameter 1)
    # and then hash that, and compare it to the user's hash
    does_match = bcrypt.checkpw(password_bytes, hash_bytes)

    return does_match
```

```
# Api's
```

```
@app.route("/", methods=["GET",
"POST"]) def login():
    if(request.method == "POST"):

        # get the data from the form
        password = request.form['password']
        email = request.form['email']

        # initialize the cursor
        # signup_cursor = mysql.connection.cursor()

        # check whether user already exists
        # user_result = signup_cursor.execute(
            "SELECT * FROM USERS WHERE user_email=%s", [email]
        )

        if(user_result > 0):
            data = signup_cursor.fetchone()
            data_password = data[3]
            if(verify_password(password,
            data_password)):
                signup_cursor.close()
                session['id'] = data[0]
                session['name'] = data[1]
                session['email'] = data[2]
                return redirect(url_for("home"))
            else:
                return render_template('login.html', error=1)
        else:
            return render_template('login.html', error=2)
    return render_template('login.html', error=3)
```

```
@app.route("/signup", methods=["POST", "GET"])
```

```

def signup():
    if(request.method == "POST"):

        # get the data from the form
        name = request.form['name']
        email = request.form['email']
        password = request.form['password']

        # hash the password
        pw_hash = create_bcrypt_hash(password)

        # initialize the cursor
        signup_cursor = mysql.connection.cursor()

        # check whether user already exists
        user_result = signup_cursor.execute(
            "SELECT * FROM USERS WHERE user_email=%s", [email]
        )
        if(user_result > 0):
            signup_cursor.close()
            return render_template('signup.html',
                error=True)
        else:
            # execute the query
            signup_cursor.execute(
                'INSERT INTO USERS(user_name,user_email,user_password,user_type)
VALUES(%s,%s,%s,%s)', (
                    name, email, str(pw_hash), "2"
                )
            )

            mysql.connection.commit()
            signup_cursor.close()
            return redirect(url_for('login'))

    return render_template('signup.html', error=False)

@app.route("/home", methods=["POST", "GET"])
def home():
    if(session['id'] == None):
        return redirect(url_for('login'))

    if(request.method == "POST"):
        # get data
        lat = request.form["lat"]
        lon = request.form["lon"]
        vis = 0
        if(lat == "" or lon == ""):

```

```

        return render_template('home.html', name=session['name'], email=session['email'],
id=session['id'], success=0)

# create a location cursor
location_cursor = mysql.connection.cursor()

# Execute the query
location_cursor.execute(
    'INSERT INTO LOCATION(location_lat,location_long,location_visited)
VALUES(%s,%s,%s)', ( lat, lon, vis
    )
)
mysql.connection.commit()
location_cursor.close()
return render_template('home.html', name=session['name'], email=session['email'],
id=session['id'], success=True)
return render_template('home.html', name=session['name'], email=session['email'],
id=session['id'])

@app.route("/logout")
def logout():
# remove the username from the session if it is there

session['id'] = None
session['name'] = None
session['email'] = None
return redirect(url_for('login'))

@app.route("/data")
def data():
if(session['id'] == None):
return redirect(url_for('login'))

location_cursor = mysql.connection.cursor()

# check whether user already exists
user_result = location_cursor.execute(
    "SELECT * FROM LOCATION"
)
if(user_result == 0):
return render_template("data.html", responses=0)
else:
res = location_cursor.fetchall()
print(res)
return render_template("data.html", responses=res)

@app.route("/android_sign_up", methods=["POST"])

```

```

def upload():
    if(request.method == "POST"):

        # get the data from the form
        name = request.json['name']
        email = request.json['email']
        password = request.json['password']

        # hash the password
        pw_hash = create_bcrypt_hash(password)

        # initialize the cursor
        signup_cursor = mysql.connection.cursor()

        # check whether user already exists
        user_result = signup_cursor.execute(
            "SELECT * FROM USERS WHERE user_email=%s", [email]
        )
        if(user_result > 0):
            signup_cursor.close()
            return {'status': 'failure'}
        else:
            # execute the query
            signup_cursor.execute(
                'INSERT INTO USERS(user_name,user_email,user_password,user_type)
VALUES(%s,%s,%s,%s)', (
                    name, email, str(pw_hash), "1"
                )
            )

            mysql.connection.commit()
            id_result = signup_cursor.execute(
                'SELECT user_id FROM USERS WHERE user_email = %s', [email]
            )
            if(id_result > 0):
                id = signup_cursor.fetchone()
                return {"id": id[0]}
            signup_cursor.close()

        return {"status": "failure"}

@app.route("/get_all_users")
def getusers():
    signup_cursor = mysql.connection.cursor()

    # check whether user already exists
    user_result = signup_cursor.execute(
        "SELECT * FROM USERS"
    )

```

```

    )
    if(user_result > 0):
        rv = signup_cursor.fetchall()
        row_headers = [x[0] for x in signup_cursor.description]
        json_data = []
        for result in rv:
            json_data.append(dict(zip(row_headers, result)))
        return json.dumps(json_data)

@app.route("/post_user_location_data", methods=["POST"])
def post_user_location():
    if(request.method == "POST"):

        # get the data from the
        formlat = request.json['lat']
        lon = request.json['long']
        id = request.json['id']
        ts = request.json['timestamp']

        # initialize the cursor
        user_location_cursor = mysql.connection.cursor()

        # execute the query
        user_location_cursor.execute(
            'INSERT INTO USER_LOCATION(location_lat,location_long,user_id,timestamp)
VALUES(%s,%s,%s,%s)', (
            lat, lon, id, ts
        )
        )

        mysql.connection.commit()

        return {"response": "success"}

@app.route("/location_data")
def location_data():
    location_cursor = mysql.connection.cursor()

    # check whether user already exists
    user_result = location_cursor.execute(
        "SELECT * FROM LOCATION"
    )
    if(user_result != 0):
        res = location_cursor.fetchall()
        print(res)
        row_headers = [x[0] for x in location_cursor.description]
        json_data = []

```



```

    for result in res:
        json_data.append(dict(zip(row_headers, result)))
    return json.dumps(json_data)
else:
    return {"response": "failure"}

@app.route("/send_trigger", methods=["POST"])
def send_trigger():
    if(request.method ==
# "POST"): get the data from
        the form email =
            request.json['email']
        location_id = request.json['id']
        location_cursor = mysql.connection.cursor()
#
        check whether user already exists
        user_result = location_cursor.execute(
            "SELECT location_visited FROM LOCATION WHERE location_id=%s", [
                location_id]
        )
        if(user_result == 0):
            return {"response": "failure"}
        else:
            res = location_cursor.fetchone()
            print(res[0])
            visited = res[0]
            visited = visited+1
            location_cursor.execute(
                "UPDATE LOCATION SET location_visited = %s WHERE location_id=%s",
                (visited, location_id)
            )
            mysql.connection.commit()
            send_mail(email)
            return {"response": "success"}

if __name__ == "__main__":
# app.run(host='0.0.0.0', port=5000)

```

DATA.HTML:

```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Zones</title>
<link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
      integrity="sha384-
Vkoo8x4CGsO3+Hhvx8T␣/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
      crossorigin="anonymous" />
<style>
  body {
    padding-top: 30px;
    padding-bottom: 30px;
    background-color: # 699cc5;
  }

  a {
    color: black;
  }
</style>
</head>

<body>
  <div class="m-4 container">
    <h1><u>Location data and Visited People</u></h1>
  </div>
  <div class="m-4 container">
    <table class="table">
      <thead>
        <tr>
          <th scope="col">S.No</th>
          <th scope="col">Latitude</th>
          <th scope="col">Longitude</th>
          <th scope="col">No_Visited</th>
        </tr>
      </thead>
      <tbody>

        {% for row in responses %}
        <tr>
          <th scope="row">{{loop.index}}</th>
          <td>{{row[1]}}</td>
          <td>{{row[2]}}</td>
          <td>{{row[3]}}</td>
        </tr>
        {% endfor %}
      </tbody>
    </table>
  </div>
  <div class="m-3 float-right">

```

```

        <button type="button" class="btn btn-danger"><a href={{url_for("home")}}>Go to location
update Page</a></button>
    </div>

</body>

</html>

```

HOME.HTML

```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

    <link
                                                rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
                                                integrity="sha384-
Vkoo8x4CGs03+Hhvx8l"/Q5PaXtkKtu6ug5l'OeNV6gBiFeWPGFN9MuhOf23Q9lfjh"
crossorigin="anonymous" />
    <style>
        body {
            padding-top: 30px;
            padding-bottom: 30px;
            background-color: #699cc5;
        }

        a {
            color: black;
        }
    </style>
</head>

<body>
    {% if success == True %}
    <script>
        alert("Location Uploaded Successfully");
    </script>
    {% elif success == 0 %}
    <script>
        alert("Enter Proper Location data");
    </script>
    {% endif %}
    <div class="m-3 float-right">

```

```

        <button type="button" class="btn btn-primary"><a href={{url_for("logout")}}>Log
Out</a></button>
    </div>
    <div class="container m-3">
        <h1><u>Declare Containment Zone</u></h1>
    </div>
    <div class="container m-3">
        <h3>welcome: {{name}}</h3>
    </div>
    <form method="POST" action="/home">
        <div class="container">
            <div class="form-group row">
                <div class="col-sm-6">
                    <label class="control-label">Lat.:</label>
                    <input type="text" class="form-control" id="lat" name="lat" />
                </div>
                <div class="col-sm-6">
                    <label>Long.:</label>
                    <input type="text" class="form-control" id="lon" name="lon" />
                </div>
                <div class="col-sm-6">
                    <label>Get current Location:</label>
                    <button type="button" class="btn btn-warning" onclick="getLocation()">Current
Location</button>
                    <label>(Click this first)</label>
                </div>
            </div>

            <!-- map -->
            <div id="map_disp" style="height: 400px; width: 500px;"></div>
            <div class="m-3 float-right">
                <button type="submit" class="btn btn-danger">Declare Containment Zone</button>
            </div>
            <div class="m-3">
                <button onclick="toggleTips()" type="button" class="btn btn-
secondary">Tutorial</button>
                <div id="tips" class="m-3">
                    <ol>
                        <li>Select The Location By Clicking the Current Location Button</li>
                        <li>Drag the Pin to change the location</li>
                        <li>Click on Declare Containment Zone to save the location to the database </li>
                    </ol>
                </div>
            </div>
            <div class="m-3 float-right">
                <button type="button" class="btn btn-warning"><a href="{{url_for('data')}}">Click Here TO
View The Containment Zones and Number of

```

```

        people visited</a></button>
    </div>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.min.js"
        integrity="sha384-
+YQ4JLhgyBLPDQt//l+S┐sc9iw4uQqACwlvpslubQzn4u2UU2UFM80nGisd026JF"
        crossorigin="anonymous"></script>
<script src="https://code.jquery.com/jquery-2.2.4.min.js"></script>
<script
src="https://maps.google.com/maps/api/js?sensor=false&libraries=places"></script>
<script
        src="https://rawgit.com/Logicify/jquery-locationpicker-
plugin/master/dist/locationpicker.jquery.js"></script>

<script>
    function getLocation() {
        if (navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(showPosition);
        } else {
            alert("No location");
        }
    }
    function showPosition(position) {
        $('#map_disp').locationpicker({
            location: {
                latitude: position.coords.latitude,
                longitude: position.coords.longitude
            },
            radius: 0,
            inputBinding: {
                latitudeInput: $('#lat'),
                longitudeInput: $('#lon'),
            },
            enableAutocomplete: true,
            onChange: function (currentLocation, radius, isMarkerDropped) {
                // Uncomment line below to show alert on each Location Changed event
                // alert("Location changed. New location (" + currentLocation.latitude + ", " +
currentLocation.longitude + ")");
            }
        });
    }
    function toggleTips() {
        var x = document.getElementById("tips");
        if (x.style.display === "none") {
            x.style.display = "block";
        } else {
            x.style.display = "none";
        }
    }

```

```
    }  
  }  
</script>  
</body>  
</html>
```

GitHub Link:

<https://github.com/IBM-EPBL/IBM-Project-29443-1660125587>

Video Demo Link:

<https://drive.google.com/uc?i>

[d=1HLXRpimSo2LmGkstJ70y](https://drive.google.com/uc?i)

[wQMknZR39LZD&export=do](https://drive.google.com/uc?i)

[wnload](https://drive.google.com/uc?i)